Artificial Intelligence Project--RLE and MIT Computation Center
Memo 35 -- LAP (LISP ASSEMBLY PROGRAM)

LAP is an internal two pass assembler for LISP 1.5. It is a pseudo-function with two arguments called the listing and the symbol table.

Before defining LAP precisely, we shall give two examples:

Example 1:

PROG2 is defined as: (LAMBDA (X Y) Y)

The LAP program for PROG2 is:

```
LAP ((  PROG2  SUBR 2) (XCA) (TRA 1 4))NIL)
```

Example 2:

At location 8763_8, the following patch is to be made:

```
                PDX  ,2
                TXL  A,2,CAR-1
                TXH  *+3,2,CAR
                CLA  (BAD ARGUMENT) in decrement of word
                TRA  ER1
            A   TRA  2,4
```

where ER1 is location 12043_{10}.

The LAP program is as follows:

```
LAP  ((  8763Q
         (PDX  0 2)
         (TXL  ((* 3) 2 ((ECAR) -1))
         (TXH  A 2  (E CAR))
         (CLA  (QUOTE (BAD ARGUMENT)))
         (TRA ER1)
       A (TRA 2 4)

     ) ((ER1 . 12043))          )
```

Definitions:

First argument of LAP: Listing

Second argument of LAP: Symtab

The listing is a one level list of items each of which is either an origin an instruction or a symbol.

The first item on the listing is always the origin.

Any item except the first is a symbol if it is atomic.

Any item except the first is an instruction if it is non-atomic.

Origin:

The first symbol on the listing is taken as the origin statement. It is interpreted as follows:

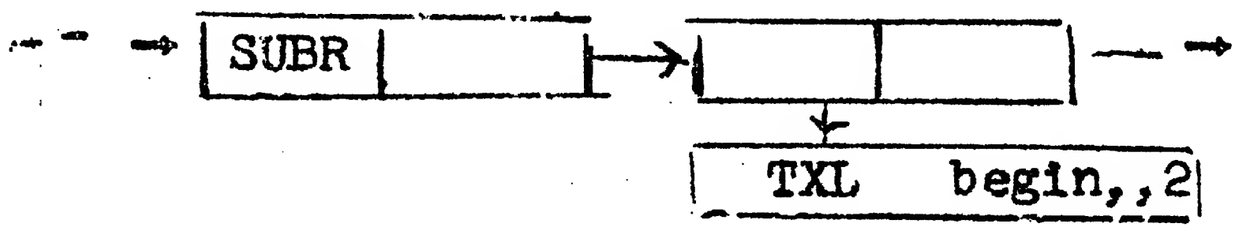
1. If the origin is a LISP number, then the assembly will begin at that location . It may be an octal number or a decimal number.

2. If the origin is an atomic symbol other than NIL, it must have a SYM on its property list or error L1 will occur. A SYM on the property list of an atomic symbol indicates a value, op-code, location, or some other quantity for LAP. The low order bits of the number pointed to by the SYM are taken as the origin. See OPDEFINE.

3. If the origin is NIL, the assembly will start at the first available location in Binary Program Space (BPS). The origin marker for BPS will be updated to avoid writing over this assembly by the next assembly. If the assembly is terminated by an error, this does not occur. If after the first pass, the assembler determines that there is not room in BPS for the entire assembly, then the second pass is cancelled and no assembly takes place. This is error condition L2.

4. If the origin is a list of the type (name type n), then the assembly is in BPS with the same protective devices as in case 3 above. After the assembly is completed, a pointer to the binary program is placed on the property list of the atomic symbol name. The pointer is flagged by the atomic symbol "type" which is usually SUBR or FSUBR. The number of arguments "n" is placed in the decrement of the pointer. The prefix contains the op-code TXL.

In example 1, the origin (PROG2 SUBR 2) put the following pointer on the property list of PROG2:



Symbols:

Atomic symbols appearing on the top level of the listing are defined as having the value of the next instruction following the symbol. Any number of symbols may appear at any point in the listing. During pass 1, a symbol table is built up. It might look like this:

((A.3426Q) (SYBOL.3425Q))

If the second argument of LAP is not NIL, it will be taken as the initial symbol table.

Instructions:

The instruction has up to four fields. Each field is evaluated separately. The result of each field evaluation is a 36 bit quantity. The fields are then assembled as follows:

1. The op field is stored at the current location.
2. The address field is reduced modulo 2^{15} and OR'ed into the address of the current location.

An arithmetic - 1 with a negative sign bit is changed to 77777.

3. The tag field is multiplied by 2^{15} and OR'ed into the current location. The instruction CLA* 1,4 is written (CLA 1 604).
4. The decrement field is reduced modulo 2^{15} and OR'ed into the decrement of the current location.

Field Evaluations:

All fields are evaluated in the same way, regardless of their position in the instruction.

If the field is an atomic, evaluation is as follows:

null[field] → contents of the cell ¤ORG. This contains the location of the next assembly in BPS, if the current assembly is not in BPS. Otherwise it contains the origin of the current assembly. This symbol is used to write patches into BPS.

eq[field;*] → current location

field $\mathcal{E}$ symbol → sassoc[field;symtab]

number ρ [field] → actual number

T → get[field;SUBR] or get[field;FSUBR] or get[field;SYM] or error[L3]

If the field is non-atomic, then the evaluation occurs in the following order.

eq[car[field];E] → cadr[field]. (E α) will evaluate to the compliment pointer to α. This appears as the address portion of the field value.

eq[car[field];QUOTE] → cons[NIL;cadr[field]].

(E ∝) will evaluate as a direct pointer to a cell containing in its decrement. This cell is put on the protected quote list. The quote list does not contain duplication.

eq[car[field];SPECIAL] → get[field;SPECIAL].

T → the sum of all of the subfields.

A non-atomic that does not begin with E,QUOTE, or SPECIAL is assumed to be a list of subfields whose values will be added together. The subfields themselves may be fields of any type except that they may not have sub-subfields.

Error Diagnostics

L1 - Unable to evaluate symbolic origin. No Assembly.

L2 - Out of Binary Program Space. Second Pass Cancelled.

L3 - Undefined symbol. Assembly incomplete.

L4 - Field with sub-subfields is illegal. Assembly incomplete.

OPDEFINE

To define new quantities for the assembler, use the pseudo-function OPDEFINE. Its argument is a list of pairs. The first member of each pair is a symbol, the second is a value,

Example:

```
OPDEFINE ((     (CLA  0500Q8)
                (TRA  0020Q8)
                (LOAD 100Q)
                (OVBGN 7432)        ))
```

The following op-codes are defined in the system,

AXT	STQ	TNZ
CLA	STR	TRA
LDQ	STZ	TSX
SLQ	SUB	TXI
STD	SXA	TZE
STO	TNX	XCA

CS-TR Scanning Project
Document Control Form

Date : 11/30/95

Report # AIM-35

Each of the following should be identified by a checkmark:
Originating Department:

☒ Artificial Intellegence Laboratory (AI)
☐ Laboratory for Computer Science (LCS)

Document Type:

☐ Technical Report (TR) ☒ Technical Memo (TM)

☐ Other:_____

Document Information

Number of pages: 4(8-IMAGES)
Not to include DOD forms, printer intstructions, etc... original pages only.

Originals are:

☐ Single-sided or

☐ Double-sided

Intended to be printed as :

☐ Single-sided or

☐ Double-sided

Print type:

☐ Typewriter ☐ Offset Press ☐ Laser Print
☐ InkJet Printer ☐ Unknown ☒ Other: MIMEOGRAPH

Check each if included with document:

☐ DOD Form ☐ Funding Agent Form ☐ Cover Page
☐ Spine ☐ Printers Notes ☐ Photo negatives
☐ Other: _____

Page Data:

Blank Pages(by page number): _____

Photographs/Tonal Material (by page number): _____

Other (note description/page number):

Description :	Page Number:
IMAGE MAP: (1-4)	1-4
(5-8) SCANCONTROL, TRGT'S(3)	

Scanning Agent Signoff:

Date Received: 11/30/95 Date Scanned: 12/1/95 Date Returned: 12/1/95

Scanning Agent Signature: Michael W. Cook

Scanning Agent Identification Target

Scanning of this document was supported in part by the **Corporation for National Research Initiatives,** using funds from the **Advanced Research Projects Agency** of the **United states Government** under Grant: **MDA972-92-J1029.**

The scanning agent for this project was the **Document Services** department of the **M.I.T Libraries.** Technical support for this project was also provided by the **M.I.T. Laboratory for Computer Sciences.**